22

Verifying Duration Properties of Timed Transition Systems

Z. Liu
Department of Mathematics and Computer Science, University of Leicester, University Road, Leicester LE1 7RH, U.K.; E-mail: *Z.Liu@mcs.le.ac.uk.*
A. P. Ravn
Department of Information Technology, Technical University of Denmark, Building 344, Lyngby, Denmark; E-mail: *apr@it.dtu.dk.*
X. Li
Faculty of Science and Technology, University of Macau, P.O. Box 3001, Macau; E-mail: *fstxsl@umac.mo.*

Abstract

This paper proposes a method for formal real-time systems development: Requirements and high level design decisions are time interval properties and are therefore specified in the Duration Calculus (DC), while implementations are described by timed transition systems (TTS). A link from implementation properties to the requirement and design properties is given by interpreting a DC formula in a model of the executions of a TTS and then providing rules for lifting properties proved by structural induction for a TTS to DC formulas. The method is illustrated by the Gas Burner case study.

Keywords

Real-time systems; Duration Calculus; Timed Transition Systems, Specification, Verification.

1 INTRODUCTION

Duration calculus (DC) was introduced to specify and reason about properties of embedded real-time systems (Zhou, Hoare & Ravn 1991, Ravn, Rischel & Hansen 1993), it has also been successfully used to reason about circuits and digital designs and hybrid systems (Ravn & Rischel 1991, Zhou, Ravn & Hansen 1993). Introductions to the applications are found in (Zhou 1993) while the logic and its foundation is given a thorough presentation in (Hansen & Zhou 1997).

A system is in DC modelled within a conventional dynamical systems framework with states that are functions of time, the non-negative reals. A Boolean state is observed through its duration in a bounded interval, i.e. the integral of an indicator function. A property for a single interval is specified by an arithmetic relation among durations and real numbers. Properties for sequences of intervals are expressed by

Programming Concepts and Methods D. Gries, & W-P. de Roever (Eds.)

the associative binary "chop" operator of the Interval Temporal Logic (Moszkowski 1985).

In a specification of requirements, the dynamical systems model is beneficial, because it focuses on relevant observable states. Furthermore, the interval logic has simple interpretations in timing diagrams for the states. The requirements will thus focus on sequencing of durational relations between observable states rather than on state transitions. A typical real-time property is *bounded critical duration*: in every interval of at most length δ the duration of a state P should not be more than γ time units. In DC this is expressed by the formula:

$$\ell \leq \delta \Rightarrow \int P \leq \gamma$$

However, when dealing with an implementation represented as a state machine such formulas do not lend themselves directly to structural induction proof methodologies. Also, at a design level, stronger assumptions may be made about timing. For example, a finite number of state changes may be assumed to occur simultaneously at a point of time, though the order in which they occur is still significant. Therefore, properties at a real-time point becomes significant when one introduces time into widely accepted transition systems models for reactive systems in general, e.g. (Keller 1976). While transition systems and the linear temporal logic (LTL), e.g. (Manna & Pnueli 1981), have been used successfully for specification and verification of reactive systems, they are not expressive enough for systems with real-time constraints, typically *bounded invariance* and *bounded responsiveness*.

Real-time is introduced into transition systems either by associating lower and upper bounds on enabled transitions (e.g. (Lamport 1977)) or by introducing explicit clocks, e.g. (Abadi & Lamport 1992). The first approach has led to extensions of LTL with bounded operators (Koymans, Vytopil & de Roever 1983), and the second has used LTL with explicit clocks, e.g. (Abadi & Lamport 1992). The relationship between the two approaches is investigated in (Henzinger, Manna & Pnueli 1994).

An advantage of TTS frameworks is that they are quite close to implementations using digital hardware and thus directly reflect semantics of programming languages. Another advantage is that safety properties and time bounded properties can be easily checked by structural induction over the transitions. However, this framework is inherently operational and less abstract than a dynamical systems based framework. For instance, properties that relate states across several transitions have to be expressed using auxiliary state variables (Lamport 1993). Thus it seems well justified to link the two formalisms such that one may use DC for high level specification and reasoning and TTS for implementation and refinement.

In a previous work (Sørensen, Hansen & Løvengreen 1994), a link is demonstrated through definition of a combination of the logics of TLT and Duration Calculus. That work investigated some of the techniques presented here. However, the current work differs by focusing on the proof system and proof techniques, which turns out to be the difficult and important part when combining theories.

After this introduction, Section 2 gives an overview of DC and its application to high level specification. Section 3 introduces TTS and a model of their executions called trajectories, which allows us to define the satisfaction of a DC formula by a

TTS in Section 4. In Section 5, we provide the rules for proving duration properties of a TTS. In Section 6, the Gas Burner example illustrates the approach and allow comparisons between this mixed approach and DC-only (Ravn et al. 1993) or TTS-only (Lamport 1993) methods.

2 REQUIREMENT AND DESIGN SPECIFICATION IN DC

A dynamic system is traditionally modelled by its state functions which are *evaluations* (or interpretations)of the system *state variables* Π over the time domain, each of them has a signature $\mathcal{V}(x) : Time \longmapsto V$, where V is the value domain of $x \in \Pi$.

The $Time$ domain must be totally ordered, and any interval $[t_1, t_2]$ between two time points is measurable by a non-negative real number called the *length* of the interval. The usual interpretation is the set of non-negative real numbers $\mathbf{R}^{\geq 0}$, but we extend it to $\mathbf{R}^{\geq 0} \times \mathbf{N}$ later in this paper.

A state function $\mathcal{V}(x)$ must be sectionally continuous and locally bounded. Sectional continuity means that in a bounded interval there is a finite number of discontinuities. For discrete state variables, which are the only variables that we use in this paper, a state function is a step function.

A *state assertion* P is a first order predicate over the state variables Π and interpreted a Boolean valued function $\mathcal{V}(P) : Time \longmapsto \{0, 1\}$; $\mathcal{V}(P)(t) = 1$ if the evaluations of the variables in P at t make P hold, and $\mathcal{V}(P)(t) = 0$ otherwise. The sectional continuity implies that $\mathcal{V}(P)$ is a step function.

The elementary observation of a state assertion P is its *duration*, written $\int P$. It denotes for a given time interval $[t_1, t_2]$, the integral $\int_{t_1}^{t_2} \mathcal{V}(P)(t)dt$. In particular, the largest duration corresponding to the length of the real-time interval is $\int 1$, which we abbreviate as ℓ. The duration is a real number and a measure. It satisfies all the properties of a measure.

Elementary formulas are arithmetic relations in durations and real valued rigid variables and constants. An example is that P holds throughout an interval: $\int P = \ell$.

The propositional connectives and quantification over rigid variables are used to generate composite formulas. An example is that P holds on a proper (non-point) interval:

$$\lceil P \rceil \stackrel{\text{def}}{=} (\int P = \ell) \wedge (\ell > 0)$$

Analogously, we introduce the abbreviation for a 'point':

$$\lceil \rceil \stackrel{\text{def}}{=} \ell = 0$$

Properties of subintervals are specified by the binary 'chop'-operator, denoted by '; '. A formula D_1 ; D_2 holds on an interval $[t_1, t_2]$ just when there exists an intermediate t such that the formula D_1 holds on $[t_1, t]$ and the formula D_2 holds on $[t, t_2]$.

The chop operator and durations are linked by the key property of summation over subintervals: for any real numbers $\gamma, \delta \geq 0$

$$(\int P = \gamma + \delta) \Leftrightarrow ((\int P = \gamma) \ ; \ (\int P = \delta))$$

For more details about the semantics, axioms and rules of DC, we refer the reader to (Hansen & Zhou 1997).

2.1 Specification of the Gas Burner

We use the Gas Burner example in (Ravn et al. 1993) to show the simplicity in writing requirement and design specifications in DC. This case study formulate the safety requirement of a gas burner in terms of a variable *Leak* denoting an undesirable but unavoidable state where unlit gas escapes from the nozzle. Safety requires that '*gas must never leak for more than 4 seconds in any period of at most 30 seconds*':

$$Req \stackrel{\text{def}}{=} \ell \leq 30 \Rightarrow \int Leak \leq 4$$

To meet the requirement *Req*, we make two design decisions:

$$Des_1 \stackrel{\text{def}}{=} \lceil Leak \rceil \Rightarrow \ell \leq 4$$
$$Des_2 \stackrel{\text{def}}{=} \lceil Leak \rceil \ ; \ \lceil \neg Leak \rceil \ ; \ \lceil Leak \rceil \Rightarrow \ell \geq 26$$

Des_1 is a bounded response property: it says that any occurrence of *Leak* must be stopped within 4 seconds. The other formula Des_2 is a bounded invariance: it says that once a *Leak* is stopped, this persists for at least 26 seconds.

The correctness of the design, i.e. implication of the validity of *Req* by that of $Des \stackrel{\text{def}}{=} Des_1 \wedge Des_2$, is shown in (Zhou et al. 1991) and for a more complex design in (Ravn et al. 1993).

The formula *Des* may be interpreted as a real-time automaton or a real-time program, which is refined further in a detailed design. However, this is in general a 'big' formula and a direct proof of refinement in DC may be tedious, because many of the steps deals with encoding of the behaviour of the automaton. We propose to deal with these steps by adopting the inductive proof techniques of the TTS framework.

3 TRANSITION SYSTEMS

This section starts with an overview of transition systems, as given in (Manna & Pnueli 1992). As in (Henzinger et al. 1994) a transition system is then extended to a timed version by imposing timing constraints on its transitions. We deviate slightly from these sources when we define the behaviour of a TTS in a trajectory model which is convenient to use as a semantic model for DC in Section 4.

3.1 Transition systems

A *transition system* $S = \langle \Pi, \Sigma, \Theta, \mathcal{T} \rangle$ consists of four components:

1. Π is a finite set of *state variables*.

2. Σ is a set of *states*. A state s in Σ assigns to every variable $u \in \Pi$ a value $s[u]$.
3. The *initial condition* Θ is a state assertion that defines a subset of Σ called the *initial states*.
4. A finite set $\mathcal{T}$ of *transitions*. Every transition τ in $\mathcal{T}$ is a binary relation on Σ; it defines for every state s, a (possibly empty) set of τ-successors $\tau(s)$. We use $en(\tau)$ to denote the enabling condition of a transition τ, and transition τ is *enabled* on a state s just when $\tau(s) \neq \emptyset$. When a τ holds between states s and s' this is often written $s \xrightarrow{\tau} s'$.

A *computation* (execution, run) of a transition system S is an infinite state sequence $\sigma = \sigma_0, \sigma_1, \cdots$ which satisfies the following two conditions:

Initiation: σ_0 satisfies Θ.

Consecution: For all $i \geq 0$, either $\sigma_i = \sigma_{i+1}$ (a *stuttering step*) or there is a transition τ in $\mathcal{T}$ such that $\sigma_i \xrightarrow{\tau} \sigma_{i+1}$ (a *diligent step*). In the later case, we say that a τ step is *taken* at position i of σ.

Thus, a computation either contains infinitely many diligent steps, or it *terminates* with an infinite stuttering-only suffix. The set $[\![S]\!]$ of all computations of S is *stuttering closed*: if an infinite state sequence σ is a computation of the program, then so is any state sequence obtained from σ by adding or deleting a finite number of stuttering steps. Stuttering closure is the key to relate system specifications (or models) (Abadi & Lamport 1991) at different level of abstractions by *refinement mappings*.

As in Manna and Pnueli's Linear Temporal Logic (LTL) (Manna & Pnueli 1992) or Lamport's Temporal Logic of Actions (TLA) (Lamport 1994), we represent transition τ of a system S as first order Boolean-valued over the state variables Π and their 'primed versions' Π', which is interpreted over pairs of states:

$\langle s, s' \rangle \models \tau(\Pi, \Pi')$ iff $\tau(s[\Pi]/\Pi, s'[\Pi]/\Pi')$ holds

3.2 Timed transition systems

We incorporate time in the transition systems models by assuming that all transitions happen "instantaneously", while timing constraints require transitions to be performed neither *too early* nor *too late*. Each transition τ is given a *lower time bound* l_τ, and an *upper time bound* u_τ. The lower time bound is a value from $\mathbf{R}^{\geq 0}$, which we take to be the set of non-negative real numbers, and the upper time bound is either a value from $\mathbf{R}^{\geq 0}$ or the special symbol ∞, which denotes the absence of an upper bound. Any real number in $\mathbf{R}^{\geq 0}$ is assumed to be less than ∞, and the lower bound is assumed not to exceed the upper bound for any transition.

A *timed transition system* $TS = \langle \Pi, \Sigma, \Theta, \mathcal{T}, l, u \rangle$ consists of an underlying transition system $S = \langle \Pi, \Sigma, \Theta, \mathcal{T} \rangle$ and two functions l and u defining the transition time interval for each transition.

3.3 Trajectories

Instead of defining a computations as a *timed state sequence* (Henzinger et al. 1994), we introduce the notion of a trajectory in order to interpret DC formulas over behaviours of a TTS. A *trajectory* over a set Π of state variables is a state function: $\rho : Time \longmapsto \Sigma$, where Σ is the set of states.

The time domain is $Time \stackrel{\text{def}}{=} \mathbf{R}^{\geq 0} \times \mathbf{N}$ in this framework. We refer to a pair $\langle t, n\rangle \in Time$ as a *position*, and $\rho(t, n)$ as the state at position $\langle t, n\rangle$ in ρ. Positions in $Time$ are ordered by the lexicographic ordering: $\langle t_1, n_1\rangle \prec \langle t_2, n_2\rangle$ iff $t_1 < t_2$ or $(t_1 = t_2) \wedge (n_1 < n_2)$. We write $\langle t_1, n_1\rangle \preceq \langle t_2, n_2\rangle$ if $\langle t_1, n_1\rangle \prec \langle t_2, n_2\rangle$ or $\langle t_1, n_1\rangle = \langle t_2, n_2\rangle$. For two positions p_1 and p_2 such that $p_1 \preceq p_2$, the *super-dense time interval* $[p_1, p_2]$ between p_1 and p_2 is the set of positions $\{p : p_1 \preceq p \preceq p_2\}$. We also define the *length* $d_1(\langle t_1, n_1\rangle, \langle t_2, n_2\rangle)$ of an interval $[\langle t_1, n_1\rangle, \langle t_2, n_2\rangle]$ as $t_2 - t_1$.

In order to make a trajectory ρ isomorphic to a timed state sequence in (Henzinger et al. 1994), we impose onto it the *sectional continuity* (or *finite variability*) condition: For any interval $[\langle a, m\rangle, \langle b, n\rangle]$, there exits a finite number of positions $\langle t_1, n_1\rangle, \cdots, \langle t_k, n_k\rangle$ such that

1. $a = t_1 < t_2 <, \cdots, < t_k = b$, and
2. ρ is constant on each interval $[\langle t_i, n_i\rangle, \langle t_{i+1}, 0\rangle]$.

Define the projection $\rho(t)$, $t \in \mathbf{R}^{\geq 0}$, as $\rho(t, 0)$. Then the above two conditions ensure that: (a) a state change can only occur between two consecutive positions $\langle t, n\rangle$ and $\langle t, n+1\rangle$, for some $t \in \mathbf{R}^{\geq 0}$ and some $n \in \mathbf{N}$; (b) the state sequence at a time point $t \in \mathbf{R}^{\geq 0}$ terminates with a constant state, denoted as $\rho(t, \infty)$; and (c) the first state $\rho(t)$ at a time point t equals the constant state of the preceding evolution: $\rho(t) = \rho(t^-)$; and the final state is the value of the succeeding evolution: $\rho(t, \infty) = \rho(t^+)$.

If $\rho(t, n) \stackrel{\tau}{\rightarrow} \rho(t, n+1)$, we say transition τ is *taken* at position $\langle t, n\rangle$ in ρ, and τ is taken at time point t in ρ if there exists a n such that τ is taken at $\langle t, n\rangle$ in ρ.

A trajectory $\rho : Time \longmapsto \Sigma$ is a *trajectory* of a timed transition system, $TS = \langle \Pi, \Sigma, \theta, \mathcal{T}, l, u\rangle$, if it satisfies the following conditions.

Initiation: $\rho(0, 0)$ satisfies Θ.

Consecution: For any position $\langle t, n\rangle$, either $\rho(t, n) = \rho(t, n+1)$ (a stuttering step) or there is a $\tau \in \mathcal{T}$ such that $\rho(t, n) \stackrel{\tau}{\rightarrow} \rho(t, n+1)$ (a diligent step).

Lower bound: if τ is taken at position p, there must exist a $p_1 \preceq p$ such that $d_1(p, p_1) \geq l_\tau$, and for any p_2 such that $p_1 \preceq p_2 \prec p$, τ is not taken.

Upper bound: if τ is enabled at position p, there exists $p_1 \succeq p$ such that $d_1(p_1, p) \leq u_\tau$ and τ is taken or is disabled at p_1.

A trajectory ρ of TS that satisfies the first two conditions is a trajectory of its un-

derlying transition system. It is easy to see that the set $[\![TS]\!]^{traj}$ of all trajectories of a system TS is isomorphic to the set of timed state sequences of TS defined in (Henzinger et al. 1994).

Intuitively, the design decisions of Des for the Gas Burner system are implemented by a two-state and two-transition TS, denoted as GB_1, which is formally described below in terms of its initial condition and transitions with their time bounds:

$$\begin{array}{lll} GB_1 = \langle & \Theta_1 : true & \\ & \tau_1 : Leak \wedge \neg Leak', & l_{\tau_1} = 0, \quad u_{\tau_1} = 4 \\ & \tau_2 : \neg Leak \wedge Leak', & l_{\tau_2} = 26, \quad u_{\tau_2} = \infty \\ \rangle & & \end{array}$$

The problem is, within a DC framework, to formally prove that GB_1 satisfies Des and then how to refine it.

4 SATISFACTION OF A DC FORMULA BY A TTS

To deal with a TTS in DC, we need to extend DC to cope with instantaneous transitions, and thus sequences of point properties. This section defines DC in terms of trajectories of transition systems and define the satisfaction relation of a duration formula by a TTS. This semantic definition preserves all the validity of the original DC axioms and rules.

4.1 Syntax

We keep the concept of a state assertion P, its duration $\int P$, rigid variables v, and define as usual an atomic duration formula F as any arithmetic relation over real valued constants, durations and rigid variables.

To describe state changes in a point interval, a transition τ, as well as an integer arithmetic relation over the *step counter* $\hbar$ and rigid variables, are also atomic formulas.

In addition to the binary "chop" modality, denoted by '; ', for specifying properties of subintervals, We introduce the unary modality $\triangleright$ which allows us to either prefixing or extending an interval.

A DC formula D is generated from the atomic formals by the Boolean connectives, the binary chop modality, the unary modality $\triangleright$, and quantification over rigid variables.

4.2 Semantics

A formula of this extended logic is to be interpreted over a trajectory ρ and a superdense interval $[p_1, p_2]$ between two positions $p_1 = \langle t_1, n_1 \rangle$ and $p_2 = \langle t_2, n_2 \rangle$. We

define two measures of such an interval by the two-valued function

$$d(p_1, p_2) \stackrel{\text{def}}{=} \begin{cases} (t_2 - t_1, n_2) & \text{if } t_1 < t_2 \\ (0, n_2 - n_1) & \text{if } t_1 = t_2 \wedge n_1 \leq n_2 \end{cases}$$

We shall use $d_1(p_1, p_2)$ and $d_2(p_1, p_2)$ to denote the first (the length of the interval) and second (the steps of the interval) values of $d(p_1, p_2)$ respectively.

For a trajectory ρ over a state space Σ, an observation over an interval $[p_1, p_2]$ consists of the states $\{\rho(t, n) \ : \ p_1 \preceq \langle t, n\rangle \preceq p_2\}$. In particular, at any *time point* t, every observation over an interval $[\langle t, n_1\rangle, \langle t, n_2\rangle]$ (which is a *point interval*) is a finite non-empty state sequence $\rho(t, n_1), \cdots, \rho(t, n_2)$. The observation of ρ over $[\langle t, n\rangle, \langle t, n\rangle]$ is the single state $\rho(t, n)$, i.e. a state sequence of 'length 0' is a single state.

Durations of states

The duration $\int P$ for a state assertion P over a super-dense interval $[\langle t_1, n_1\rangle, \langle t_2, n_2\rangle]$ is still used to define the accumulated time when P holds within the super-dense interval:

$$\int_{t_1}^{t_2} P(\rho(t, 0))dt$$

Thus, a duration ignores all the discontinuous positions, and its existence is guaranteed by the finite variability of the trajectory*. Under these definitions, all the conventional DC axioms in (Zhou et al. 1991) remain valid.

State properties and transitions

A state property P which is a first order predicate assertion (no quantification is allowed over state variables) holds for a trajectory ρ over an observation if it holds at the first position of the observation:

$$\rho, [\langle t_1, n_1\rangle, \langle t_2, n_2\rangle] \models P \quad \text{iff} \quad \rho(t_1, n_1) \models P$$

A transition τ is treated as an interval formula as for the case of a state predicate:

$$\rho, [\langle t_1, n_1\rangle, \langle t_2, n_2\rangle] \models \tau \quad \text{iff} \quad \langle \rho(t_1, n_1), \rho(t_1, n_1 + 1)\rangle \models \tau$$

Obviously, a state predicate is a special transition which does not refer to primed variables, and all axioms in the first order predicate logic remain valid for reasoning about state properties and transitions.

The composite formula $\lceil\ \rceil \wedge (\hbar = 0) \wedge \tau$ specifies the point observations containing a single state in which τ is enabled; and the formula $\lceil\ \rceil \wedge (\hbar > 0) \wedge \tau$ defines the point observations consisting of a sequence of more than one state such that the first two of them is a τ-step.

The Hoare Triple $\{P\}\tau\{Q\}$ for any state assertions P and Q and any state transition τ will be used in structural induction reasoning, and it is defined here as the

*The logic remains the same if $\int P$ is defined as $\int_{t_1}^{t_2} \forall n \in \mathbf{N} \bullet P(\rho(t, n))dt$ or as $\int_{t_1}^{t_2} \exists n \in \mathbf{N} \bullet P(\rho(t, n))dt$.

state transition

$$\{P\}\tau\{Q\} \stackrel{\text{def}}{=} P \wedge \tau \Rightarrow Q'$$

where Q' is obtained from Q by replacing each state variable x of Q by its primed version x'.

The length of a state sequence in an interval is given by the step counter $\hbar$:

$$\rho, [p_1, p_2] \models \hbar = n \quad \text{iff} \quad d_2(p_1, p_2) = n$$

The bounded version of the temporal formula $\bigcirc D$ (i.e. the 'next' operator) can be defined in terms of $\lceil\ \rceil$, $\hbar$ and the chop operator as $(\lceil\ \rceil \wedge \hbar = 1)\ ;\ D$.

Modalities

The 'chop' operator now chops a super-dense interval. A formula $D_1\ ;\ D_2$ holds on such an interval $[p_1, p_2]$ just when there exists an intermediate position p such that the formula D_1 holds on $[p_1, p]$ and the formula D_2 holds on $[p, p_2]$:

$$\rho, [p_1, p_2] \models D_1\ ;\ D_2 \text{ iff}$$
$$\exists p \bullet (p_1 \preceq p \preceq p_2) \wedge (\rho, [p_1, p] \models D_1) \wedge (\rho, [p, p_2] \models D_2)$$

A formula $\triangleright D$ holds on an interval $[b, e]$ there is an interval $[b, f]$ which is either a prefix or an extension of $[b, e]$ such that D holds on $[b, f]$:

$$\rho, [b, e] \models \triangleright D \quad \text{iff} \quad \exists f \succeq b \bullet (\rho, [b, f] \models D)$$

This operator distributes over disjunction and is monotone. It comes as a combination $\Diamond_l \Diamond_r$ of the two neighbourhood modalities $\Diamond_l$ and $\Diamond_r$ from (Zhou & Hansen 1996*a*). In this paper, we are only concerned about future properties which can be treated sufficiently by the combined operator. Axioms and properties for the chop operator and the $\triangleright$ operator (treated as $\Diamond_l \Diamond_r$) given in (Zhou et al. 1991) and (Zhou & Hansen 1996*a*) remain valid under the new interpretation.

With operator $\triangleright$, the *conventional linear temporal formulas* $\Diamond D$ and $\Box D$ can be defined as follows $\triangleright(true\ ;\ D)$ and $\neg\Diamond\neg D$, respectively.

A trajectory ρ *satisfies* a formula D if for any $\langle r, n\rangle \in Time$ $\rho, [\langle 0, 0\rangle, \langle r, n\rangle] \models D$. *Valid* formulas are satisfied by any trajectory. A formula D is *valid for* (or *satisfied by*) a TTS, TS, denoted by $TS \models D$, iff any trajectory $\rho \in [\![TS]\!]^{traj}$ satisfies D.

4.3 Proof System

We have just said that in the extended logic, the validity of the axioms and rules in the original DC in (Moszkowski 1985, Zhou et al. 1991, Zhou & Hansen 1996*a*) for durations and the two modalities is preserved as well as the validity of the axioms and rules in first predicate logic is for state properties and transitions at a time point. For example, it is easy to see that as in original DC and ITL, 'chop' is associative, distributes over disjunction, and has *false* as zero. Furthermore, for any durational formula D, $\lceil\ \rceil$ is the unit of 'chop': $\lceil\ \rceil\ ;\ D \Leftrightarrow D\ ;\ \lceil\ \rceil \Leftrightarrow D$. However, we need

axioms to link point properties with interval properties. The following axioms about temporal variable $\hbar$ play this role:

A1 :	$\exists n : \mathbf{N} \bullet (\hbar = n)$	
A2 :	$(D \Leftrightarrow \lceil\,\rceil \wedge (\hbar = 0)\ ;\ D) \wedge (D \Leftrightarrow D\ ;\ \lceil\,\rceil \wedge (\hbar = 0))$	unit of chop
A3 :	$\lceil\,\rceil \wedge (\hbar = n > 0) \Leftrightarrow (\lceil\,\rceil \wedge \hbar = 1\ ;\ \lceil\,\rceil \wedge (\hbar = n-1))$	
	$\lceil\,\rceil \wedge (\hbar = n > 0) \Leftrightarrow (\lceil\,\rceil \wedge \hbar = n-1\ ;\ \lceil\,\rceil \wedge (\hbar = 1))$	chop a point
A4 :	$(\ell = r) \Rightarrow \exists n \bullet (\lceil\,\rceil \wedge (\hbar = n)\ ;\ (\ell = r))$	super-dense
A5 :	$\neg(\lceil\,\rceil \wedge \hbar = 0) \Rightarrow ((\{P\}\tau\{Q\}) \Leftrightarrow (P \wedge \tau \Rightarrow \bigcirc Q))$	link positions

For a state predicate P we use $P^\bullet$ to denote that P holds *anywhere* in a interval:

$$P^\bullet \stackrel{\text{def}}{=} \neg(true\ ;\ \neg P\ ;\ true)$$

5 LIFTING PROPERTIES OF TTS TO DURATION PROPERTIES

The 'yardstick' properties of a timed transition system include *bounded invariance properties*, and *bounded progress properties*. These properties and the structural induction proof rules for them are given in (Henzinger et al. 1994) in timed linear temporal logic (TLTL). This section shows how these properties are incorporated into the extended DC framework.

A state property P is an *invariant* of a transition system S if P holds at any position in any interval for all trajectories of S. Thus, an unbounded invariance property is of the form $\Box P$, where P is a state predicate. P is an invariant of a transition system S iff P is implied by Θ, and P is preserved by all transitions in $\mathcal{T}$. Let $\{P\}\mathcal{T}\{Q\}$ hold iff $\mathcal{T}\{P\}\tau\{Q\}$ holds for each $\tau \in \mathcal{T}$. An induction rule for proving an invariant of S is given in (Manna & Pnueli 1981) as:

UB-INV $$\frac{\Theta \Rightarrow P,\ \{P\}\mathcal{T}\{P\}}{\Box P}$$

which is still sound in the interval logical setting, i.e. any transition system satisfying the premises also satisfies the conclusion.

There are other induction rules for proving untimed properties which we leave out of this paper, because our goal is the time bounded properties. We are mainly concerned with *bounded properties*.

As in (Henzinger et al. 1994), we are interested in proving bounded-invariance and bounded-response properties, and thus restrict ourselves to the following bounded temporal formulas:

Primitive formulas: State formulas and state transitions are (*linear*) *temporal formulas*.

Boolean connectives: Every Boolean combination of temporal formulas is a temporal formula.

Bounded-eventually formulas: If ϕ is temporal formula and $u \in \mathbf{R}^{\geq 0}$, then so is $\Diamond_{\leq u}\phi$; it is true over a trajectory ρ with an initial position $\langle t_0, n_0\rangle$ iff there is a position $p = \langle t, n\rangle$ such that $t \leq t_0 + u$ and ϕ holds for the suffix of ρ starting with

p. We thus define

$$\Diamond_{\leq u}\phi \stackrel{\text{def}}{=} (\ell < u) \vee (\ell \leq u\ ;\ \phi) \quad \text{or equivalently}$$
$$\Diamond_{\leq u}\phi \stackrel{\text{def}}{=} \rhd(\ell \leq u\ ;\ \phi)$$

Bounded-unless formulas: If P is a state properties, ϕ is a temporal formula, and $l \in \mathbf{R}^{\geq 0}$, then $P\,\mathcal{U}_{\geq l}\phi$ is a temporal formula; it is true over a trajectory ρ iff P holds at the initial position $\langle t_0, n_0\rangle$ of ρ and either P holds for all positions in ρ, or there there is some position p such that $t \geq t_0 + l$, and ϕ holds for the suffix starting with p, and P holds at every position $q \prec p$. This formula can thus be defined as

$$P\,\mathcal{U}_{\geq l}\phi \stackrel{\text{def}}{=} (P^{\bullet} \vee (P^{\bullet} \wedge (\ell \geq l)\ ;\ \bigcirc\phi))$$

We use the convention that the letters P, Q, R, as well as φ, denote state formulas; τ and its indexed versions denote state transitions; ϕ, ψ and χ stand for temporal formulas; and D, F and their indexed versions, denote general formulas in the extended logic. We also denote $P\,\mathcal{U}_{\geq l}true$ by $\Box_{<l}P$ when $l > 0$, and define $\Box_{<0}P$ to be $true$.

5.1 Induction rules for bounded properties

We start with *bounded invariance properties*. A bounded invariance property asserts that when some state occurs a condition holds continuously for a certain amount of time. Such a property is specified by a temporal formula of the form $P \Rightarrow \Box_{<l}Q$. Therefore, to prove a bounded invariance property, we often have to prove a *bounded unless* property. There are five main rules for bounded unless in (Henzinger et al. 1994) given in Table 1, which are still sound in our setting.

$\mathcal{U}$-**MON**	$\dfrac{P \Rightarrow P_1,\quad \phi \Rightarrow \phi_1,\quad l_1 \leq l}{(P\,\mathcal{U}_{\geq l}\phi) \Rightarrow (P_1\,\mathcal{U}_{\geq l_1}\phi_1)}$
$\mathcal{U}$-**TRAN**	$\dfrac{\varphi \Rightarrow P\,\mathcal{U}_{\geq l_1}\chi,\quad \chi \Rightarrow Q\,\mathcal{U}_{\geq l_2}\psi}{\varphi \Rightarrow (P \vee Q)\,\mathcal{U}_{\geq l_1+l_2}\psi}$
$\mathcal{U}$-**SS**	$\dfrac{P \Rightarrow \neg en(\tau),\quad P \Rightarrow \varphi,\quad \varphi \Rightarrow Q,\quad \{\varphi\}\mathcal{T}-\tau\{\varphi\},\quad \{\varphi \wedge en(\tau)\}\tau\{R\}}{P \Rightarrow Q\,\mathcal{U}_{\geq l_\tau}R}$
$\mathcal{U}$-**CSS**	$\dfrac{P \Rightarrow \neg en(\tau),\quad \{Q\}\mathcal{T}-\tau\{\neg R\}}{P\,\mathcal{U}_{\geq l}Q\,\mathcal{U}\,(R \wedge \phi) \Rightarrow P\,\mathcal{U}_{\geq l}Q\,\mathcal{U}_{\geq l_\tau}(R \wedge \phi)}$
$\mathcal{U}$-**COLL**	$(P\,\mathcal{U}_{\geq l_1}Q\,\mathcal{U}_{\geq l_2}\phi) \Rightarrow ((P \vee Q)\,\mathcal{U}_{\geq l_1+l_2}\phi)$

Table 1 Rules for bounded unless

A *bounded progress property* asserts that once P occurs, Q must occur within a certain amount of time, and thus is specified in the form $P \Rightarrow \Diamond_{\leq u}P$. Again, the rules for bounded progress properties in (Henzinger et al. 1994) can be brought into our framework, which are given in Table 2.

$\Diamond$-**MON**	$\dfrac{\phi \Rightarrow \phi_1, \quad u_1 \geq u}{\Diamond_{\leq u}\phi \Rightarrow \Diamond_{\leq u_1}\phi_1}$
$\Diamond$-**TRAN**	$\dfrac{\phi \Rightarrow \Diamond_{\leq u_1}\chi, \quad \chi \Rightarrow \Diamond_{\leq u_2}\psi}{\phi \Rightarrow \Diamond_{\leq u_1+u_2}\psi}$
$\Diamond$-**SS**	$\dfrac{P \Rightarrow (\varphi \vee Q), \quad \varphi \Rightarrow en(\tau), \quad \{\varphi\}\mathcal{T} - \tau\{\varphi \vee Q\}, \quad \{\varphi\}\tau\{Q\}}{P \Rightarrow \Diamond_{\leq u_\tau} Q}$
$\Diamond$-**CSS**	$\dfrac{P \Rightarrow en(\tau), \quad \{P\}\mathcal{T}\{\neg P\}}{P\,\mathcal{U}\,\phi \Rightarrow \Diamond_{\leq u_\tau}\phi}$
$\Diamond$-**COLL**	$(P_1 \Rightarrow \Diamond_{\leq l_1}\phi) \wedge (P_2 \Rightarrow \Diamond_{\leq l_2}\phi) \Rightarrow ((P_1 \vee P_2) \Rightarrow \Diamond_{\leq max\{l_1,l_2\}}\phi)$

Table 2 Rules for bounded progress

5.2 Linking rules

Now we are ready to provide the rules which link temporal properties to the DC-*invariance* properties, DC-*bounded response* properties, and the DC-*minimal separation* properties. The rules are presented in the form such that the premises of a rule are temporal formulas (including first order predicates) and the conclusion is an formula in DC. This indicates that an establishment of a DC property can be achieved from some TLTL properties which can be proven by the structural induction rules.

The DC-invariance property $\lceil\,\rceil \vee \lceil P \rceil$ can also be derived from the temporal invariance property $\Box P$:

LINK-1 $\dfrac{\Box P}{\lceil\,\rceil \vee \lceil P \rceil}$

A DC-bounded response property is of the form $\neg(\lceil P \rceil \;;\; (\lceil \neg Q \rceil \wedge \ell > u))$, asserting that P must be followed within u time units by an occurrence of Q, and can be proved by the following linking rule:

LINK-2 $\dfrac{\begin{array}{l} P \Rightarrow \Diamond_{\leq u} Q \\ \neg Q \Rightarrow \neg Q\,\mathcal{U}\,Q\,\mathcal{U}_{\geq l}\neg Q, \quad \text{for some } l > 0 \end{array}}{\neg(\lceil P \rceil \;;\; (\lceil \neg Q \rceil \wedge \ell > u))}$

A *minimal separation* property is described as a DC-formula of the form

$$\neg((\lceil P \rceil \;;\; \lceil Q \rceil) \wedge (\ell < l) \;;\; D \;;\; true)$$

which asserts that D cannot hold until Q has been stable for at least l time units since the change from a P-state to a Q-state took place. Using bounded unless properties, we can prove a minimal separation property by the following link:

LINK-3 $\dfrac{\begin{array}{l} Q \Rightarrow \neg P, \quad \phi \Rightarrow \neg Q \\ \psi \Rightarrow P\,\mathcal{U}_{\geq l_1}\,Q\,\mathcal{U}\,\phi \\ P\,\mathcal{U}_{\geq l_1}\,Q\,\mathcal{U}\,\phi \Rightarrow P\,\mathcal{U}_{\geq l_1}\,Q\,\mathcal{U}_{\geq l_2}\phi \end{array}}{(\psi \;;\; true) \Rightarrow \neg((\lceil P \rceil \;;\; \lceil Q \rceil) \wedge (\ell < l_1 + l_2) \;;\; \phi \;;\; true)}$

This rule is inductive in the sense that ϕ and ψ are temporal formulas which may be linked further to DC-formulas. The first two premises $Q \Rightarrow \neg P$ and $R \Rightarrow \neg Q$

ensure the pattern $(\lceil P \rceil \ ; \ \lceil Q \rceil \ ; \ \phi)$ does not collapse, the premise in the second line says that a ψ-state sequence pattern can only proceed with P lasting for at least l_1 time units followed by a Q-state and then a ϕ pattern, the premise in the third line then ensures that Q lasts for at least l_2 time units.

The last two premises in the rule may be combined into one. However, the one in the second line have to be established which then can be used to establish the last one by rule $\mathcal{U}$ -CSS. The special case when $l_1 = 0$ and ψ is P is often used. In this case the rule becomes

LINK-3′
$$\frac{\begin{array}{l} Q \Rightarrow \neg P \qquad R \Rightarrow \neg Q \\ P \Rightarrow P \,\mathcal{U}\, Q \,\mathcal{U}\, R \\ P \,\mathcal{U}\, Q \,\mathcal{U}\, R \Rightarrow P \,\mathcal{U}\, Q \,\mathcal{U}_{\geq l} R \end{array}}{(\lceil P \rceil \ ; \ \lceil Q \rceil \ ; \ \lceil R \rceil) \Rightarrow (\ell \geq l)}$$

6 IMPLEMENTATION AND REFINEMENT OF THE GAS BURNER

This section shows how the linked methods are used in the development of the Gas Burner. We first show that the timed transition system GB_1 given in Section 3.2 satisfies the specification *Des* in Section 2.1. Then we show how GB_1 is *refined* into an implementation that may have only ignition failure. Finally, we present an implementation of *Des* which tolerates both ignition and flame failures.

Theorem 1 *The timed transition system GB_1 given in Section 3.2, satisfies Des_1 and Des_2.*

This theorem follows from the following lemma:

Lemma 1 *The time transition system GB_1 satisfies the following TLTL properties:*

(T_1) : $Leak \Rightarrow \Diamond_{\leq 4} \neg Leak$
(T_2) : $Leak \Rightarrow Leak \,\mathcal{U}\, \neg Leak \,\mathcal{U}\, Leak$
(T_3) : $Leak \Rightarrow Leak \,\mathcal{U}\, \neg Leak \,\mathcal{U}_{\geq 26} Leak$

Proof: (T_1) is proven by applying $\Diamond$-**SS** to $\tau_1 \stackrel{\text{def}}{=} Leak \wedge \neg Leak'$ and $u_{\tau_1} = 4$, with the following premises:

$Leak \Rightarrow en(\tau_1)$ as $en(\tau_1) = Leak$
$\{Leak\}\tau_2\{Leak\}$ as $Leak \wedge \neg Leak \wedge Leak' \Rightarrow Leak'$
$\{Leak\}\tau_1\{\neg Leak\}$ as $Leak \wedge Leak \wedge \neg Leak' \Rightarrow \neg Leak'$

(T_2) is obviously valid in the untimed LTL proof system. From (T_2), we then have (T_3) by applying $\mathcal{U}$ -**CSS** for τ_2 with $l_{\tau_2} = 26$. ♡

Proof of Theorem 1: Des_1 is deduced by **LINK-2** with (T_1) and (T_3) as the premises. Des_2 follows from **LINK-3′** by letting P, Q, and R in the rule be $Leak$, $\neg Leak$ and

Leak, respectively, to establish the first two premises of the rule; while (T_2) and (T_3) are the last two premises.

An implementation without flame failure

After the initial implementation, GB_1 can be refined in the traditional TTS framework. For example, the transition system GB_2 in Figure 1 is a refinement of GB_1.

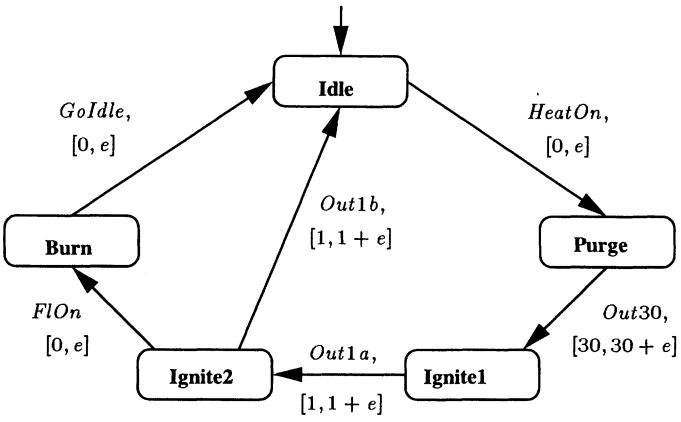

Figure 1 A refinement of GB_1

GB_2 has the following *phases*:

Idle: Await heat request, no gas and ignition. It enters **Purge** within e time units on heat request. The constant parameter e in this example is the system wide upper bound for *reacting**.

Purge: Pauses for 30 seconds and then enters **Ignite1** within e time units.

Ignite1: Turns on ignition and gas supply and enters after one second the **Ignite2** phase within e.

Ignite2: Monitors the flame and enters **Burn** if it is sensed within 1 second, otherwise returns to **Idle** while turning the gas off within e.

Burn: Ignition is switched off but gas is still on. The **Burn** phase is stable until heat request goes off. Gas is then turned off and **Idle** is entered within e.

We use a simple error recovery: return to **Idle** from **Ignite2**. We assume no flame failure in the **Burn** phase. Therefore, in this implementation, *Leak* can only occur in the **Ignite1** and **Ignite2** phases.

*In (Ravn et al. 1993, Lamport 1993), a lower bound is also given. It is not needed for proving the correctness of this design.

With the convention that the value of a variables x is changed by a transition τ only if τ names x', the formal definition of this refined transition systems is as follows.

$$
\begin{array}{lll}
GB_2 = \langle \ \Pi & \stackrel{\text{def}}{=} & \{gas, ignition, sensor, phase\} \\
\Theta & \stackrel{\text{def}}{=} & (gas, ignition, sensor, phase) = (\text{off}, \text{off}, \text{off}, \textbf{Idle}) \\
HeatOn & \stackrel{\text{def}}{=} & HeatReq \wedge (phase = \textbf{Idle}) \wedge (phase' = \textbf{Purge}) \\
Out30 & \stackrel{\text{def}}{=} & (phase = \textbf{Purge}) \wedge \\
 & & (phase', gas', ignition') = (\textbf{Ignite1}, \text{on}, \text{on}) \\
Out1a & \stackrel{\text{def}}{=} & (phase = \textbf{Ignite1}) \wedge (phase' = \textbf{Ignite2}) \wedge \\
 & & (sensor' = \text{on}) \\
Out1b & \stackrel{\text{def}}{=} & (phase = \textbf{Ignite2}) \wedge (phase' = \textbf{Idle}) \wedge \\
 & & (ignition' = \text{off}) \wedge (gas' = \text{off}) \\
FlOn & \stackrel{\text{def}}{=} & (phase = Ignite2) \wedge Flame \wedge (phase' = \textbf{Burn}) \wedge \\
 & & (ignition' = \text{off}) \\
GoIdle & \stackrel{\text{def}}{=} & (phase = \textbf{Burn}) \wedge \neg HeatReq \wedge (phase' = \textbf{Idle}) \wedge \\
 & & (gas' = \text{off}) \wedge (sensor' = \text{off}) \\
\rangle &
\end{array}
$$

The time bounds are given by Figure 1.

Let $Leak_1$ hold iff $phase = \textbf{Ignite1}$ holds and $Leak_2$ hold iff $phase = \textbf{Ignite2}$ holds. In the following, we present the properties of these two *leak* states of GB_2 which *correspond* to the properties of *Leak* and have same proof routines as those for GB_1.

Lemma 2 *The timed transition system GB_2 satisfies the following TLTL properties provided $e \leq 1$:*

	Property of GB_2	*Property of* GB_1
(T_{11}) :	$Leak_1 \Rightarrow \Diamond_{\leq 1+e} \neg Leak_1$	T_1 of *Leak*
(T_{12}) :	$Leak_1 \Rightarrow Leak_1 \,\mathcal{U}\, \neg Leak_1 \,\mathcal{U}\, Leak_1$	T_2 of *Leak*
(T_{13}) :	$Leak_1 \Rightarrow Leak_1 \,\mathcal{U}\, \neg Leak_1 \,\mathcal{U}\, _{\geq 31} Leak_1$	T_3 of *Leak*
(T_{21}) :	$Leak_2 \Rightarrow \Diamond_{\leq 1+e} \neg Leak_2$	T_1 of *Leak*
(T_{22}) :	$Leak_2 \Rightarrow Leak_2 \,\mathcal{U}\, \neg Leak_2 \,\mathcal{U}\, Leak_2$	T_2 of *Leak*
(T_{23}) :	$Leak_2 \Rightarrow Leak_2 \,\mathcal{U}\, \neg Leak_2 \,\mathcal{U}\, _{\geq 31} Leak_2$	T_3 of *Leak*

By the induction rules, Lemma 2 leads to the following corollary:

Corollary 1 (GB_2 **refines** GB_1:) *Let* $Leak \stackrel{\text{def}}{=} Leak_1 \vee Leak_2$ *and* $e \leq 1$. GB_2 *satisfies the properties* (T_1), (T_2) *and* (T_3) *in Lemma 1.*

Thus, we have the following theorem corresponding to Theorem 1 for GB_1.

Theorem 2 *Let* $Leak \stackrel{\text{def}}{=} Leak_1 \vee Leak_2$ *and* $e \leq 1$. GB_2 *satisfies* Des_1 *and* Des_2, *as well as Req, of the Gas Burner system.*

Lemma 3 *The timed transition system* GB_2 *has the following properties if* $e \leq 1$*:*

	Property of GB_2	*Property of* GB_1
(Des_{11}) :	$\lceil Leak_1 \rceil \Rightarrow \ell \leq 1 + e$	Des_1 of $Leak$
(Des_{12}) :	$\lceil Leak_1 \rceil \ ; \ \lceil \neg Leak_1 \rceil \ ; \ \lceil Leak_1 \rceil \Rightarrow \ell \geq 31$	Des_2 of $Leak$
(Des_{21}) :	$\lceil Leak_2 \rceil \Rightarrow \ell \leq 1 + e$	Des_1 of $Leak$
(Des_{22}) :	$\lceil Leak_2 \rceil \ ; \ \lceil \neg Leak_2 \rceil \ ; \ \lceil Leak_2 \rceil \Rightarrow \ell \geq 31$	Des_2 of $Leak$

Using Lemma 2, the proofs of Des_{11} and Des_{21} are the same as the proof of Des_1 of GB_1, while the proofs of Des_{12} and Des_{22} are the same as that of Des_2 for GB_1, in Theorem 1.

Using the linking rules as in the proof of Theorem 1, Lemma 3 has the corollary below:

Corollary 2 *The timed transition system* GB_2 *has the following properties if* $e \leq 1$*:*

	Property of GB_2	*Property of* GB_1
(Req_{11})	$\ell \leq 30 \Rightarrow \int Leak_1 \leq 1 + e$	Req of $Leak$ in GB_1
(Req_{21})	$\ell \leq 30 \Rightarrow \int Leak_2 \leq 1 + e$	Req of $Leak$ in GB_1

It is interesting to notice that *Req* of the Gas Burner (which is already deduced from Corollary 1 in Theorem 2) can be also deduced from Corollary 2 and the valid formula

$$\int (Leak_1 \vee Leak_2) = \int Leak_1 + \int Leak_2 = 2 + 2e \leq 4$$

An implementation with flame failure

When flame may disappear in the phase **Burn**, we can add another simple error recovery procedure which enters **Idle** from **Burn** by switching off the gas. Formally, we only have to redefine the action *GoIdle* in GB_2 as

$$\begin{aligned} GoIdle \stackrel{\text{def}}{=} \ & (phase = \mathbf{Burn}) \wedge (\neg Flame \vee \neg HeatReq) \wedge (phase' = \mathbf{Idle}) \\ & \wedge (gas' = \text{off}) \wedge (sensor' = \text{off}) \end{aligned}$$

Let $Leak_3$ denote $(phase = \mathbf{Burn}) \wedge \neg Flame$. The corresponding timed transition system GB_3 satisfies Corollary 2 and the following property

$$(Req_{31}) : \quad \ell \leq 30 \Rightarrow \int Leak_3 \leq e$$

which is of the same form as *Req* for the abstract state *Leak* in GB_1 and can be proved by the same proof routine by which *Req* is proved for GB_1.

Let *Leak* be the disjunction $Leak_1 \vee Leak_2 \vee Leak_3$. By the valid formulas

$$\begin{aligned} \int Leak &= \int (Leak_1 \vee Leak_2 \vee Leak_3) \\ &= \int Leak_1 + \int Leak_2 + \int Leak_3 = 2 + 3e \end{aligned}$$

this gives the following correctness theorem for GB_3:

Theorem 3 (**Correctness of** GB_3) *Assume that* $e \leq 2/3$, *the transition system* GB_3 *satisfies the requirement Req of the Gas Burner system.*

However, we should mention that GB_3 is not a refinement of GB_1 or GB_2. It does not implement Des as it does not meet Des_2. In other words, we do not have a corollary for GB_3 corresponding to Corollary 1 or a theorem for GB_3 corresponding to Theorem 2.

7 CONCLUSION

We have proposed an approach to combine the DC approach and the TLTL approach to specification and verification of embedded real-time systems. The combination provides both the advantage of DC for directly modelling conventional dynamical systems with state that are functions of time, and the advantage of TTS for modelling computations in reactive systems. A central point is that the link between the approaches are formulated as syntactical rules

Within this framework, DC is used for specifying the system requirements and an initial design which intuitively corresponds to a simple TTS. Then we can refine (e.g. from GB_1 to GB_2) or adjust (e.g. from GB_2 to GB_3) the initial design within the TTS framework until we obtain an implementation which is a program.

The approach is illustrated by solving the Gas Burner example. The advantage of the combined approach becomes obvious if we compare with the solution in the single DC framework in (Ravn et al. 1993), and the solution in the single TLTL framework in (Lamport 1993): specifications of the system at different levels are simpler and better structured; the proofs of the correctness of the systems at different levels are easier as they directly reflect the structural induction of each step in the refinement/implementation, and *concrete* states (e.g. $Leak_1$, $Leak_2$ and $Leak_3$ in GB_2 and GB_3) at a lower level are reasoned about in the same way as the corresponding *abstract* state (e.g. $Leak$ in GB_1) at a higher level.

The combined logic allows us to describe multiple instantaneous transitions at a time point. For the same purpose, papers (Zhou & Hansen 1996*b*) and (Xu 1997) introduce a *dense-chop operator* which can define the meaning of the sequential composition of state transitions. However, in both papers the intermediate states of a sequentially composite statement are hidden, and thus the semantics of $x := 1;\ x := x+2$ is the same as that of $x := 3$. This is a nice property if the approaches are used to deal with concurrent systems with no shared variables. However, in shared memory based models such as TTSs, the semantics of $(x := x + 1;\ x := x + 2) \parallel (x := 0)$ should be quite different from that of $x := 3 \parallel x := 0$; therefore we define DC over trajectories of a TTS.

Further work includes development of a full combined logic and extension of it to deal with hybrid transition systems (Manna & Pnueli 1993).

Acknowledgements We wish to thank Xu Qiwen for the comments and discus-

sions about the finite variability of a trajectory, and the anonymous referees for their comments.

REFERENCES

Abadi, M. & Lamport, L. (1991). The existence of refinement mapping. *Theoretical Computer Science* **83**(2), 253–284.

Abadi, M. & Lamport, L. (1992). An old-fashioned recipe for real-time. *In* J. de Bakker, C. Huizing, W. de Rover & G. Rozenberg, eds, Real-Time: Theory in Practice, Lecture Notes in Computer Science 600, Springer-Verlag, pp. 1–27.

Hansen, M. & Zhou, C. (1997). Duration calculus: logical foundations. *Formal Aspects of Computing* **9**(3), 283–330.

Henzinger, T., Manna, Z. & Pnueli, A. (1994). Temporal proof methodologies for timed transition systems. *Information and Computation* **112**(2), 273–337.

Keller, R. (1976). Formal verification of parallel programs. *Communication of the ACM* **19**(7), 371–384.

Koymans, R., Vytopil, J. & de Roever, W.-P. (1983). Real-time programming and asynchronous message passing. *In* Proc. 2nd Annual Symposium on Principles of Distributed Computing, ACM press, pp. 187–197.

Lamport, L. (1977). Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering* **3**(2), 125–143.

Lamport, L. (1993). Hybrid systems in TLA$^+$. *In* R. Grossman, A. Nerode, A. Ravn & H. Rischel, eds, Hybrid Systems, Lecture Notes in Computer Science 736, Springer-Verlag, pp. 77–102.

Lamport, L. (1994). The temporal logic of actions. *ACM Transactions on Programming Languages and Systems* **16**(3), 872–923.

Manna, Z. & Pnueli, A. (1981). The temporal framework for concurrent programs. *In* R. Boyer & J. Moore, eds, The Correctness Problem in Computer Science, Academic Press, pp. 215–274.

Manna, Z. & Pnueli, A. (1992). *The Temporal Logic of Reactive and Concurrent Systems: Specification*, Springer-Verlag.

Manna, Z. & Pnueli, A. (1993). Verifying hybrid systems, *In* R. Grossman, A. Nerode, A. Ravn & H. Rischel, eds, Hybrid Systems, Lecture Notes in Computer Science 736, Springer-Verlag, pp. 4–35.

Moszkowski, B. (1985). A temporal logic for multilevel reasoning about hardware. *IEEE Computer* **18**(2), 10–19.

Ravn, A. & Rischel, H. (1991). Requirements capture for embedded real-time systems. *In* Proc. IMACS-MCTS'91 Symposium on Modelling and Control of Technical Systems, Vol 2, pp. 1147–152.

Ravn, A., Rischel, H. & Hansen, K. (1993). Specifying and verifying requirements of real-time systems. *IEEE Transactions on Software Engineering* **19**(1), 41–55.

Sørensen, M., Hansen, O. & Løvengreen, H. (1994). Combining temporal specific-

ation techniques. *In* Temporal Logic ICTL 94, Lecture Notes in Artificial Intelligence 827, Springer-Verlag, pp. 1–16.

Xu, Q. (1997). A semantics and verification of extended phase transition systems in duration calculus, *In* O. Maler, ed., International Workshop on Hybrid and Real-Time Systems, Lecture Notes in Computer Science 1201, Springer-Verlag, pp. 301–315.

Zhou, C. (1993). Duration calculi: an overview, *In* D. Bjørner, M. Broy & I. Pottosin, eds, Proc. Formal Methods in Programming and Their Application, Lecture Notes in Computer Science 735, Springer-Verlag, pp. 256–266.

Zhou, C. & Hansen, M. (1996*a*). An adequate first order interval logic. Technical Report 91, UNU/IIST, P.O. Box 3058, Macau.

Zhou, C. & Hansen, M. (1996*b*). Chopping a point. *In* J. He, J. Cooke & P. Wallis, eds, Proc. BCS FACS 7th Refinement Workshop: Theory and Practice of System Design.

Zhou, C., Hoare, C. & Ravn, A. (1991). A calculus of durations. *Information Processing Letters* **40**(5), 269–276.

Zhou, C., Ravn, A. & Hansen, M. (1993). An extended duration calculus for hybrid real-time systems. *In* R. Grossman, A. Nerode, A. Ravn & H. Rischel, eds, Hybrid Systems, Lecture Notes in Computer Science 736, Springer-Verlag, pp. 36–59.

BIOGRAPHIES

Zhiming Liu is a lecturer in Computer Science at the University of Leicester. He obtained his MSc in Computer Science from the Software Institute of the Chinese Academy of Sciences (Beijing, 1987) and Ph.D in Computer Science from the University of Warwick (UK, 1991). His research interest is in Theories of Distributed Real-Time Systems. He has worked and had long visits to the University of Warwick and the Technical University of Denmark.

Anders P. Ravn is a Reader at the Technical University of Denmark (DTU). He works on methods for engineering of embedded real-time systems based on coherent mathematical theories and techniques. He holds a MSc in Computer Science from the University of Copenhagen (1973) and is dr. techn from DTU (1995). He has worked in industry and had long visits to IBM T.J. Watson Research Center, Oxford University, and Kiel University.

Xiaoshan Li is an assistant professor of Software Engineering at University of Macau. He received his PhD degree at Institute of Software, the Chinese Academy of Sciences (Beijing, 1993). He worked in the University of Newcastle upon Tyne from October 1995 to March 1998. His research interests include formal methods in software engineering, hardware verification, specification of concurrent, real-time and hybrid systems.